DOCUMENT RESUME

ED 250 178 SE 045 154

AUTHOR Dalbey, John; Linn, Marcia
TITLE Spider World: A Robot Language for Learning to Program. Assessing the Cognitive Consequences of Computer Environments for Learning (ACCCEL).
INSTITUTION California Univ., Berkeley. Lawrence Hall of Science.
SPONS AGENCY National Inst. of Education (ED), Washington, DC.
PUB DATE 84
GRANT 400-83-0017
NOTE 29p.; Paper presented at the Annual Meeting of the American Educational Research Association (68th, New Orleans, LA, April, 1984).
PUB TYPE Reports - Research/Technical (143) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Cognitive Processes; Computer Graphics; *Computer Software; *Independent Study; Instructional Material Evaluation; Problem Solving; *Programing; *Skill Development; Student Behavior; Transfer of Training

ABSTRACT

Spider World is an interactive program designed to help individuals with no previous computer experience to learn the fundamentals of programming. The program emphasizes cognitive tasks which are central to programming and provides significant problem-solving opportunities. In Spider World, the user commands a hypothetical robot (called the "Spider") to create colored patterns on the computer display screen. This study: (1) assessed the effectiveness of Spider World in fostering autonomous learning among students; (2) determined if students acquired the ability to use three Spider World templates ("repeat...until," "define...end," and "if...go to") in a novel situation; and (3) compared the program's impact to two other instructional conditions (using Type Attack software and instruction in BASIC). Findings indicate that using Spider World to introduce programming results in student acquisition of templates which they can apply in related environments; in contrast, students receiving instruction in BASIC appear to have made fewer gains in these skills. In addition, students exhibited their autonomous learning behaviors when given the opportunity to use them in Spider World. (JN)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED250178

Spider World: A Robot Language for Learning to Program

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
✓ This document has been reproduced as received from the person or organization originating it.
Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

John Dalbey
Marcia Linn

Assessing the Cognitive Consequences of
Computer Environments for Learning (ACCCEL)
Lawrence Hall of Science
University of California, Berkeley, CA
94720

This material is based upon research supported by the National Institute of Education under Grant No. 400-83-0017. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Institute of Education.

This paper was presented at the annual meeting of the American Educational Research Association New Orleans, 1984

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Marcia C. Linn
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

SE045154

Introduction

The purpose of this study was to assess the impact of Spider World, a robot language, as an introduction to programming. The Spider World program was recently field tested as part of a study by the NIE funded Assessing the Cognitive Consequences of Computer Environments for Learning project (ACCCEL) at the Lawrence Hall of Science. We wanted to determine if students would find it easier to learn programming if they had an initial experience using this carefully designed piece of educational software. We speculated that the experience with this software would help students learn specific thinking skills which are necessary in programming, and learn to work with the computer autonomously.

There are two major difficulties with using a general purpose programming language in an introductory course. The first is that the complexity and detail of a general purpose language like BASIC or Pascal often overwhelms the novice. The student spends most of the time learning an immense number of superficial syntax rules, and fails to grasp essential concepts of algorithm design. The second problem is that applications of a general purpose language are seldom very appealing or motivating. The kinds of problems which can be solved after an initial introduction to a general purpose language are unattractive and of little interest to the average person.

To overcome these two problems, a special purpose pro-

oramming environment called Spider World was developed (Dalbey, 1983a). Spider World is an interactive simulation for people with no previous computer experience to learn the fundamentals of programming. Spider World is similar to other "robot" languages such as turtlegraphics (Papert, 1980), Karel (Pattis, 1981), and Josef (Tomek, 1982). In Spider World the user commands a hypothetical robot called the Spider to create colored patterns on the computer display screen.

Spider World has many advantages over general purpose languages which makes it appropriate for a beginning programming course. Perhaps the most significant is that the Spider World environment is graphically oriented. Novices find the environment intrinsically appealing because it allows them to interact with and control a visual display. General purpose languages manipulate symbols or perform numeric computations, neither of which has the powerful appeal of graphics. In this regard Spider World is approachable by people who are not comfortable with "mathematical" problems.

The second advantage of Spider World is the language has a limited yet powerful command set with a very simple syntax. The commands provide both looping and subroutine capabilities. Spider instructions are simple one word english commands. There are no restrictions on punctuation such as line numbers or statement separators. There are no

mysterious abbreviations or mnemonic codes and no algebraic expressions.

The third advantage of Spider World is that there is no input and output of data. This is important because it avoids many confusing and rather esoteric issues such as reserving memory locations or declaring variables, formatting of data, obeying data type conventions, variable naming rules, and data streams. Because Spider programs have no input and output, the learner can focus on control structures for looping and subroutines. Thus, the more central ideas of procedure specification can be introduced while avoiding the peripheral issues of data manipulation.

What is unique about Spider World?

Spider World emphasizes the cognitive tasks which are central to programming, and provides experience with that kind of problem solving. There are some important distinctions between Spider and other robot languages which make her especially appropriate for beginning programming. The primary difference is that Spider World is much simpler than other robots because it offers a more limited set of capabilities. This narrow domain is advantageous because novices can immediately begin exploring the environment with little or no instruction. It means that there are not a lot of commands to learn before one encounters the central ideas of programming. This enables one to engage in cognitively demanding tasks which present significant problem solving

opportunities.

One of the unique features of Spider World is that it has a mode of operation in which the user can interactively guide the Spider to create designs on the screen. In "draw mode" the display shows a white square with black grid lines, which is the Spider's room (see Figure 1). The Spider herself is displayed in the upper left corner as a solid black triangle. From this initial position the user can "pilot" the Spider, directing her to move around the room and paint squares certain colors. The bottom four lines of the screen describe which key on the keyboard causes which action. This enables the user to begin interacting immediately without reading a user manual or text. Each button pressed causes the Spider to respond with a specific action, and results in an immediate change in the image on the screen. This beginning interactive mode of Spider World allows a novice to explore the primitive capabilities of the Spider in a very direct way. Draw mode allows the user to discover the effects of Spider commands without creating and executing a complete program.

An attractive characteristic of Spider World is that the Spider operates in a discrete world, not a continuous one like turtle graphics. The Spider functions in a plane which is divided into a small number of distinct cells which appear on the screen as a grid. The only movement available to the Spider is within this orthogonal framework. This is

especially advantageous for children who don't yet understand geometry, because it is not necessary to specify a numeric value of an angle in order to turn the Spider.

One of the most powerful features of Spider World is that the Spider remembers each key that the user presses in draw mode. Then by pressing a special key ("redraw") the entire pattern can be recreated. Thus, the simulation has created a stored program automatically, which the user can execute by simply pressing a key. This exemplifies the essential characteristic of programmable devices -- the ability to autonomously carry out a stored sequence of instructions. In this way the fundamental power of the computer is made available to the novice almost effortlessly. By providing this concrete experience with a robot, the learner has a tangible basis for understanding the abstract concept of a programmable device.

Figure 2 shows the display at a particular point during the "redraw" or "run" of a Spider program. The simulation allows the user to halt the execution, or proceed one step at a time. Included in the text at the bottom of the screen is the Spider instruction currently being executed.

The operation of the Spider in draw mode is based on a "one button per function" model which is common to most mechanical devices, such as household appliances, as well as contemporary video games. The earlier stages of interacting with the Spider are not unlike previous experiences most

students have had in daily life. The novice does not need any prerequisite knowledge from mathematics or formal languages. Thus, for even unsophisticated learners, Spider World serves as an accessible bridge to the "powerful ideas" contained in programming.

The Spider World simulation has a built-in editing facility. Simply pressing a key causes the commands in the Spider's memory to be printed as English text on the screen (see Figure 3). The user's program can be modified using certain keys to move the cursor on the screen and insert and delete lines of code. Additionally, the simulation has the ability to load and save Spider programs the user creates on a floppy disk. Thus, the user is quickly acquainted with the utilities that are commonly offered for developing and manipulating programs (see Figure 4).

The Spider language has several programming constructs which are found in higher level languages. The DEFINE-END construct provides the ability to create subroutines. The REPEAT-UNTIL construct performs a looping function. The Spider also has a "counter" and a "memory register" which can be manipulated by the program. The introduction of these powerful programming structures is facilitated by the simplicity of the command language (see Figure 5). With these structures the user can solve complex problems which demonstrate essential techniques of algorithm design.

Advantages for Programming Instruction

As we said at the onset, Spider World responds to two major difficulties presented by most programming languages. First, Spider World allows for more precise instruction in the specific concepts of algorithm design. Spider World emphasizes two flow of control features and provides students with the opportunity to fully understand these features and to use them effectively.

Students gain what we refer to as "templates" for these features. By templates, we mean stereotypic prescriptions for a particular aspect of a program. These templates are similar to the schemata described by Norman, Gentner, and Stevens (1976). Templates can apply to a whole program. For example, they can be reflected in an "input-process-output" template. Templates can also apply to a specific function of a program. In Spider World, the two templates which are specifically emphasized are: first, the "repeat...until" template; and second, the "define...end" template. The "repeat...until" template is a looping template; the "define...end" template is a subroutine template. The Spider World environment teaches these templates and provides extensive experience in using them such that students are likely to build an effective cognitive structure for these templates which they can use in other situations.

The second difficulty which Spider World is specifically designed to overcome has to do with student interest

in and willingness to pursue the learning environment. Spider World is an appealing environment, and it is designed such that students may proceed to learn autonomously. Students are encouraged to depend on the manual and to use feedback from the computer to figure out solutions to their problems, rather than depending on the instructor and using the machine solely as a typewriter.

Thus, Spider World was implemented specifically to help students acquire two templates and to foster autonomous learning among students.

Methods

Instructional Treatments

To assess the effectiveness of Spider World in imparting these two skills, we implemented Spider World for three weeks and compared the impact of Spider World to two other instructional conditions. One of the conditions was Type Attack, a software program by Sirius Software. This was chosen on the suggestion of several classroom teachers who complained that lack of typing ability is the largest obstacle to success in their programming courses. In addition, we scheduled a control group which began BASIC instruction from the start of the course. Each of these three programs were implemented for three weeks. For the Spider World and Type Attack programs, curriculum materials were supplied by

ACCCEL (Dalbey, 1983c; Nelson, note 2). These materials included lecture notes, sample problems, student exercises and worksheets, and solutions. BASIC was taught in the manner used by the teacher prior to our investigation.

Subjects

These three treatments were implemented in two suburban and one urban junior high school. One school had three classes which were randomly assigned to the three conditions. One had six classes; two of which were randomly assigned to each condition. The third school had two classes which were randomly assigned to either BASIC or Spider World. Approximately twenty-seven students were enrolled in each class.

Teacher Training

To familiarize teachers with the Type Attack and Spider World materials, a half day in-service session was scheduled. Teachers were presented with the materials, allowed to use them on-line, given the curriculum guide, and instructed to follow the curriculum guide closely. Since the teachers would not be expert in using the Spider World software, we suggested that they encourage the students to figure out answers to their questions on their own as much as possible. Students were given access to the Spider World Reference Manual and encouraged to look up answers to their questions when necessary. Teachers were encouraged to

demonstrate their own problem solving processes when questions which they could not answer were asked by students. We anticipated that teachers would engage in problem solving and test the Spider World environment when students asked them questions about aspects of the language with which they were not familiar with.

The teachers were confident, after the half-day in-service, that they could implement the program and, indeed, were very appreciative of our efforts to, in their view, help them improve their total instructional program. Although the teachers were enthusiastic about the materials that we presented, they reserved judgment concerning whether or not the materials would foster cognitive accomplishments in their students.

Assessment

After three weeks of instruction, a written assessment of student achievement was administered. The first part of this assessment focused on the specific software the student had used. Thus, we assessed typing speed in the Type Attack program; we assessed knowledge of Spider World in the Spider World program; and we assessed knowledge of BASIC in the BASIC program. The second part of the assessment was a test of template knowledge given to all participants in the study. This "robot" programming test was intended to measure the template students had acquired during the three weeks of instruction. The test had three parts. One part

focused specifically on the "repeat...until" template. A second part focused on the "define...end" or subroutine template, and a third part focused on the template which was emphasized in the BASIC instruction: the "if...goto" template. We hypothesized that students who had been exposed to the Spider World templates would have acquired the ability to use those templates in a novel situation, such as that afforded by the "robot" test. Furthermore, we hypothesized that students who had participated in BASIC would have acquired the ability to use the one template taught during their three-week lesson in a novel environment. Sample items from the robot test are given in Figures 6 and 7.

This study is part of a larger investigation. Analysis of tests of competency with the specific software used and performance after additional instruction in BASIC following these three-week introductions will be reported in subsequent publications. This report focuses exclusively on the robot test assessment of performance after three weeks of instruction.

To evaluate the effects of Spider World on student autonomous learning, in-class observations of students as well as interviews with teachers were used. Observations focused on the behavior of individual students. Student behavior was coded according to the activity engaged in (designing, coding, running, reformulating, etc.), stra-

tegies for debugging (hypothesizing, rerunning, listing, etc), and help-seeking (text, peer, teacher, etc). Analysis focused primarily on narrative comments from the observers. Each class was observed at least twice during the three weeks of instruction.

Results

Language Acquisition

Students in the BASIC treatment learned very little about the BASIC language during the first three weeks. On the exam, students were not even proficient at identifying syntactically correct BASIC statements. They were somewhat able to comprehend the operation of simple programs using PRINT, but did not understand looping or subroutines.

Students in the Spider World treatment learned quite a lot about Spider programming during the three weeks of instruction. Results of the exam showed them to be adept with the primitives of the language, and also moderately capable of understanding programs with loops and subroutines.

Template Acquisition

To assess the effect of instruction on template acquisition, the robot test was divided into three sub-

scales, each focusing on one template (subroutines, loops, or branches). Table 1 shows the relative performance of each treatment group on each subscale of the robot test.

Students in the Spider World treatment group did the best on the robot test. Students in this group had the highest means on 21 out of 24 items on the robot test. A statistically significant difference between the means for Spider World and BASIC was found on seven items, and also on the total test means.

Students in the Type Attack treatment group did better than the BASIC group on the first two sections of the robot test. These sections dealt with subroutines and loops. The Type Attack means were higher on 15 out of 16 items, with a statistically significant differences on 2 items. The BASIC group had slightly higher means than Type Attack on 7 out of 8 items in the third section, which dealt with branching.

Thus it appears that Spider World students benefited the most from their instruction. They did well on their achievement test and also outperformed the other groups on the transfer test.

The surprising result is the mediocre performance by students in the BASIC group. They did poorly on their achievement test, and were outscored on 2 out of 3 sections on the transfer test by students who had used only typing software. It appears that the small amount of BASIC that was

learned by those students did not appreciably affect their performance on the robot test. The only section in which they did better than the Type Attack group was the section that involved branching constructs, analogous to "GOTO" in BASIC.

Autonomous Learning

Assessment of the effect of the Spider World instruction on autonomous learning is based on observations and interviews with the teachers. Observers of students using Spider World reported considerably more instances of students interrogating the environment and testing it to find out how to do something more effectively. In addition, students in the Spider World environment were more likely to use reference materials than those studying BASIC. Furthermore, students in Spider World clearly enjoyed the software and frequently expressed interest in continuing to work with it.

Their teachers echoed this sentiment, requesting permission to use Spider World in their other classes and to use Spider World in subsequent years of instruction. In general, both students and teachers were very enthusiastic about Spider World, thus indicating that Spider World met the motivational criteria which we set out to investigate. More research would be helpful to further clarify whether students gained autonomous learning skills by working in

this environment. It seems clear that students exhibit their autonomous learning behaviors when given the opportunity to use them in Spider World.

Teachers also reported that when the BASIC instruction was instituted for students who had had Spider World that those classes were quick to pick up on BASIC. Indeed, the teachers reported that the Spider World classes caught up with the BASIC classes by the time that the semester was over. The teachers were unsure exactly what to attribute this to. Besides the Spider World instruction, there was perhaps an implicit expectation on the part of the teacher that it would be easier to teach classes where all were covering the same material. Thus it is not obvious whether an implicit desire to have the classes catch up with the other classes or an explicit advantage of Spider World contributed to this situation. Nevertheless, Spider World may help students to acquire further programming understanding and it seems unlikely that Spider World interferes with further learning.

Conclusions

Many observers of use of computers in pre-college settings report that little which could be defined as cognitively demanding is occurring in these classrooms. In contrast, using Spider World to introduce programming results

in student acquisition of templates which they can apply in related environments. Thus, students acquire some of the powerful ideas which have been advertised for computer learning environments and demonstrate that they can use them in new domains. In contrast, students receiving instruction in BASIC appear to have made fewer gains in these skills even though their instruction focused on a single template while the Spider World instruction focused on two.

The ACCCFL Project Staff (Linn and Fisher, 1982) have argued that autonomous learning skills might be acquired from computer learning environments. Yet, few environments offer these opportunities. It appears that the Spider World software and curriculum encouraged students to learn autonomously. More work is needed to determine whether autonomous learning can be fostered more generally in programming courses and to establish which specific features of the computer environment foster these accomplishments.

The Spider World software and curriculum were extremely well-received by teachers and students. The programming environment appears to be enhanced by use of this sort of introduction. In particular, the limited number of commands in the environment appear to help students focus on the cognitively interesting aspects of interaction with a computer. The design of the software, such that a large amount of feedback can be gained and the opportunity to use the reference manual when problems arise both contribute to the

likelihood that students will engage in autonomous learning in the computer environment.

Technical Information

Spider World runs on an Apple II-plus microcomputer with a single disk drive and 48K RAM. A color monitor is recommended though not essential. The simulation includes the interpreter, editor, file manager, and a demonstration mode. The source code is largely written in Applesoft BASIC with minor segments in machine language. An extensive reference manual and teacher's guide has been prepared (Dalbey, 1983b; Dalbey, 1983c). The simulation and documentation is available for fifteen dollars from Project ACCCEL, Lawrence Hall of Science, University of California, Berkeley, CA 94720.

Reference Notes

1. Reese, P., and Stein, J. Rocky's Boots curriculum. Project ACCCEL, Lawrence Hall of Science, 1983.

2. Nelson, C. Tyon Attack curriculum. Project ACCCEL, Lawrence Hall of Science, 1983.

References

Dalbey, J. (1983a). Spider World [Computer program]. Berkeley, CA: University of California, Lawrence Hall of Science.

Dalbey, J. (1983b). Spider World Reference Manual [Computer program manual]. Berkeley, CA: University of California, Lawrence Hall of Science.

Dalbey, J. (1983c). Spider World Teacher's Guide [Computer program manual]. Berkeley, CA: University of California, Lawrence Hall of Science.

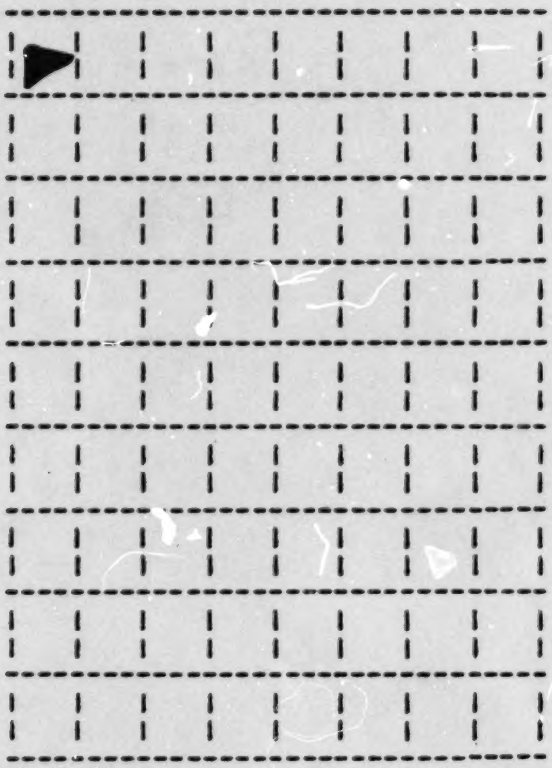
Linn, M.C. & Fisher, C. (1982). Computer education: The gap between promise and reality. Proceedings of Making Our Schools More Effective: A Conference for Educators, Far West Laboratory, San Francisco.

Norman, D., Gentner, D., & Stevens, A. (1976). Comments on learning schemata and memory representation. In D. Klahr (Ed.). Cognition and Instruction. Hillsdale, NJ: Lawrence Erlbaum Associates.

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. New York: Basic Books.

Pattis, R. (1981). Karel the robot. New York: Wiley & Sons.

Tomek, I. (1982). Josef, the robot. Computers in Education, 6, 287-293.



TELL THE SPIDER WHAT TO DRAW

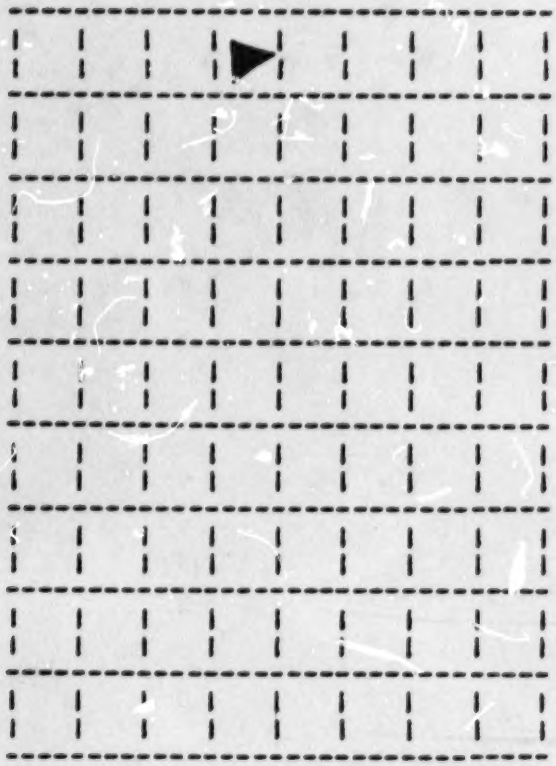
S=STEP, T=TURN, R=RED, B=BLUE, G=GREEN

P=PURPLE, W=WHITE, X=BLACK, !=REDRAW,

F=FORGET, <ESC>=LIST, Q=QUIT

Draw Mode Screen Display

Figure 1



STEP

PRESS 'H' TO HALT. SPACE BAR TO RESUME.

<ESC> TO STOP THE SPIDER.

Run Mode Screen Display

Figure 2

```
: HERE IS A SAMPLE SPIDER PROGRAM
START
BLUE
STEP
RED
TURN
STEP
GREEN
QUIT
```

```
:=UP    /=DOWN    @=INSERT    *=DELETE
OPTIONS: LIST,RUN,DRAW,FORGET,SAVE,LOAD,
         BUILD,DEMO,FILES,!,<ESC>
```

List Mode Screen Display

Figure 3

LIST: Displays the current spider program. You may make changes to the program by retyping lines on the screen. You may move the cursor to the desired line by pressing ":" for up and "/" for down. To insert a line press "=". To delete a line press "*".

RUN: Execute the current spider program.

DRAW: An interactive mode where you guide the Spider to create patterns. In draw mode the Spider is moved by pressing certain keys.

FORGET: Erase all the lines in the current program.

SAVE: Copy the current program to a disk file.

LOAD: Copy a file from the disk into the Spider memory. The old program is lost, and the program on the file becomes the current program.

BUILD: Append a file from the disk to the Spider's memory. The old program is not destroyed.

DEMO: Automatically run a demonstration of the spider in action. Demo executes a sequence of prewritten spider programs.

FILES: Displays the names of spider programs which have been saved on the disk.

Spider World Simulation Options

Figure 4

Action commands

START: Draw the wall and prepare to begin your journey.
STEP: Take one step in the direction you are facing.
TURN: Turn to your right.
BLUE: Paint the square you are standing on blue.
(The Spider also knows RED, GREEN, PURPLE, WHITE, and BLACK.)
ADD: Add one to your counter.
SUBTRACT: Subtract one from your counter.
STORE: Place the current value of the counter in memory register.
RECALL: Replace the value of counter with the value from memory register.
QUIT: Stop. your journey is over.

Control commands

DEFINE task-name Allows a sequence of statements to be referred
. to by a single name. When task-name is used
. in the program, all the statements defined
. in the task will be carried out.
END (A task must be defined before it can be invoked.

REPEAT Tells the spider to keep doing the instructions that
. are written between the "repeat" and "until" statements
. until the indicated condition is met.
UNTIL [WALL/ZERO/color]
WALL means until the wall is reached.
ZERO means until the counter equals zero.
color means until the Spider is standing on a square of the specified color.

Spider Language

Figure 5

Table 1

Robot Test Results

Item	Mean BASIC	TA	SW	F Ratio from oneway analysis of variance	Significance level
Subroutines					
1	4.4	4.8	4.5	3.4	.04
2	8.3	8.5	8.6	0.3	ns
3	6.2	7.5	7.2	3.2	.04
4	5.6	5.9	6.6	1.8	.18
5	6.4	7.4	7.6	3.6	.03
6	4.2	4.3	5.5	3.5	.03
Total	34.0	38.1	37.2	1.9	.15
Loops					
1	.45	.70	.67	6.2	.00
2	.56	.68	.74	2.7	.07
3	.61	.65	.67	.34	ns
4	.38	.40	.48	.71	ns
5	.46	.50	.59	1.3	ns
6	.30	.39	.58	6.0	.00
7	3.9	4.8	6.5	6.8	.00
8	.42	.42	.59	2.6	.07
9	.25	.35	.56	7.9	.00
10	2.6	3.5	4.5	4.6	.01
Total	9.6	12.2	14.8	5.3	.00
Branches					
1	.22	.15	.30	2.0	.13
2	.28	.25	.36	1.5	.22
3	.25	.15	.38	4.5	.01
4	.20	.14	.29	2.1	.11
5	.22	.14	.27	1.7	.18
6	.11	.07	.22	3.3	.04
7	2.6	2.3	4.0	3.3	.04
8	1.5	1.6	2.7	2.7	.07
Total	5.3	4.8	8.0	2.8	.06
All Items					
	49.0	55.2	60.0	3.2	.04

Figure 6

BEST COPY AVAILABLE

INSTRUCTIONS

Garfield Robot can learn new commands. New commands combine already known commands.

To teach Garfield Robot a new command to make a square, you could say:

```
New Command SQUARE
Move 3
Turn R
Move 3
Turn R
Move 3
Turn R
Move 3
End Command
```

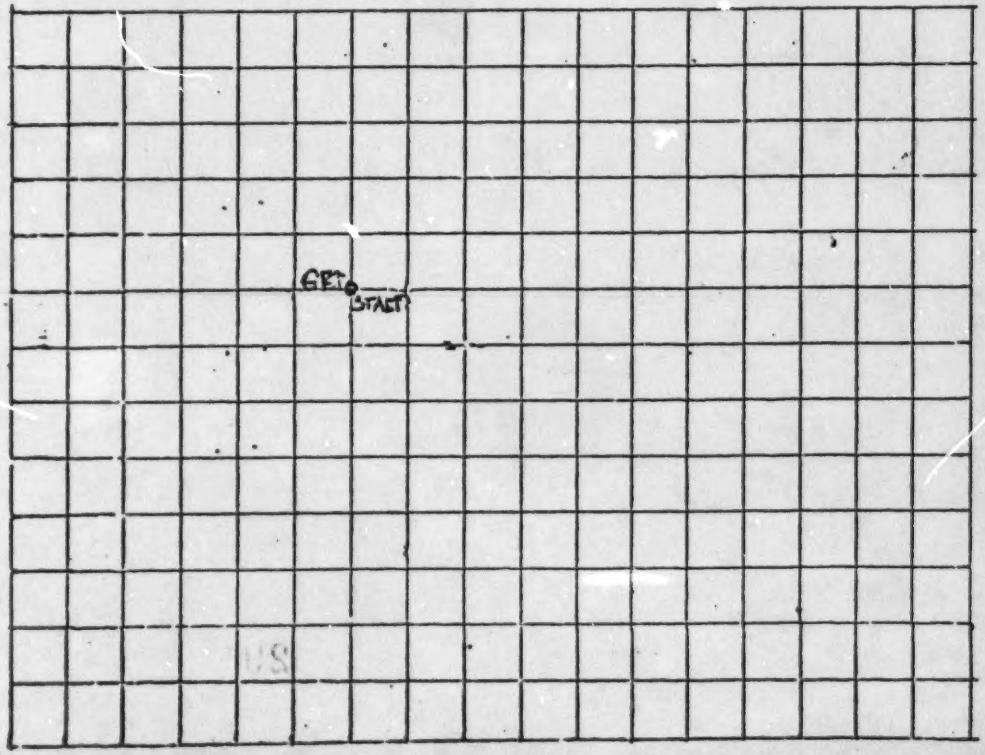
PROBLEMS

3) New commands can be used just like other commands. Draw what Garfield Robot would do to follow these commands:

COMMANDS

```
Begin
Turn R
Move 5
Square
Move 5
End
```

PATH



BEST COPY AVAILABLE

Figure 7

4) Write a program using SQUARE to get Garfield Robot to follow this path. Start where you see the GR:

COMMANDS

PATH